PROGRAM CODE

```
#include <stdio.h>
int
block[20],process[20],isAllocated[20]={0},alloc
ated[20],b,p,choice,flag = 0;

void firstFit ();
void bestFit ();
void worstFit ();
int bwFinder (int startIndex, int pSize, int
curValue, int mode);
void display ();
int main ()
{
  printf ("Number of Blocks = ");
  scanf ("%d", &b);
  printf ("Size of Each Block\n");
  for (int i = 0; i < b; i++)
    {
      printf ("B[%d] - ", i);
      scanf ("%d", &block[i]);
    }
  printf ("\nNumber of Process = ");
  scanf ("%d", &p);
  printf ("Enter the size of each Process\n");
  for (int i = 0; i < p; i++)
    {
      printf ("P[%d] - ", i);
      scanf ("%d", &process[i]);
    }
  while (choice != 4)
    {
      printf ("\n1.First Fit\n2.Best Fit\n3.Worst
Fit\n4.Exit\n");
      printf ("Choice >>> ");
      scanf ("%d", &choice);
      switch (choice)
          {
          case 1:
            {
              firstFit ();
              display ();
              break;
            }
          case 2:
            {
              bestFit ();
              display ();
              break;
            }
          case 3:
            {
              worstFit ();
              display ();
              break;
            }
          case 4:
            {
              break;
            }
          default:
            {
              printf ("\nInvalid choice");
            }
          }
    }
}

void display ()
{
  printf ("\nProcess\t\tProcess size\t\tBlock
Allocated\n");
  for (int i = 0; i < p; i++)
    {
      if (allocated[i] == -999)
          {
            printf ("%d\t\t%d\t\t\tNot allocated", i,
process[i]);
            printf ("\n");
          }
      else
          {
            printf ("%d\t\t%d\t\t\t%d", i, process[i],
allocated[i]);
            printf ("\n");
          }
    }
}

void firstFit ()
{
  for (int i = 0; i < b; i++)
    {
      isAllocated[i] = 0;
    }
  for (int i = 0; i < p; i++)
    {
      flag = 0;
      for (int j = 0; j < b; j++)
```

```
        {
          if (process[i] <= block[j] &&
isAllocated[j] == 0)
            {
              allocated[i] = block[j];
              isAllocated[j] = 1;
              flag = 1;
              break;
            }
        }
    if (flag == 0)
        {
          allocated[i] = -999;
        }
   }
}

void bestFit ()
{
  for (int i = 0; i < b; i++)
    {
     isAllocated[i] = 0;
    }
  for (int i = 0; i < p; i++)
    {
     int small = 0;
     for (int j = 0; j < b; j++)
        {
          if (process[i] <= block[j] &&
isAllocated[j] != 1)
            {
              small = block[j];
              isAllocated[j] = 1;
              small = bwFinder (j, process[i],
small, 1);
              allocated[i] = small;
              break;
            }
        }
    if (small == 0)
        {
          allocated[i] = -999;
        }
   }
}

void worstFit ()
{
  for (int i = 0; i < b; i++)
    {
     isAllocated[i] = 0;
    }
  for (int i = 0; i < p; i++)
    {
     int big = 0;
     for (int j = 0; j < b; j++)
        {
          if (process[i] <= block[j] &&
isAllocated[j] != 1)
            {
              big = block[j];
              isAllocated[j] = 1;
              big = bwFinder (j, process[i], big, 2);
              allocated[i] = big;
              break;
            }
        }
    if (big == 0)
        {
          allocated[i] = -999;
        }
   }
}

int bwFinder (int startIndex, int pSize, int
curValue, int mode)
{
  int lastBlock = startIndex;
  for (int i = startIndex + 1; i < b; i++)
    {
     if (pSize <= block[i] && (mode == 1 ?
block[i] < curValue : block[i] >
                                curValue) &&
isAllocated[i] != 1)
        {
          isAllocated[lastBlock] = 0;
          lastBlock = i;
          curValue = block[i];
          isAllocated[i] = 1;
        }
    }
  return curValue;
}
```

OUTPUT

```
administrator@administrator-hcl-desktop:~/Desktop/gopikrishna$ gcc alloc.c
administrator@administrator-hcl-desktop:~/Desktop/gopikrishna$ ./a.out
Number of Blocks = 5
Size of Each Block
B[0] - 20
B[1] - 100
B[2] - 40
B[3] - 200
B[4] - 10

Number of Process = 4
Enter the size of each Process
P[0] - 90
P[1] - 50
P[2] - 30
P[3] - 40

1.First Fit
2.Best Fit
3.Worst Fit
4.Exit
Choice >>> 1

Process         Process size            Block Allocated
0               90                      100
1               50                      200
2               30                      40
3               40                      Not allocated

1.First Fit
2.Best Fit
3.Worst Fit
4.Exit
Choice >>> 2

Process         Process size            Block Allocated
0               90                      100
1               50                      200
2               30                      40
3               40                      Not allocated

1.First Fit
2.Best Fit
3.Worst Fit
4.Exit
Choice >>> 3

Process         Process size            Block Allocated
0               90                      200
1               50                      100
2               30                      40
3               40                      Not allocated

1.First Fit
2.Best Fit
3.Worst Fit
4.Exit
Choice >>> 4
administrator@administrator-hcl-desktop:~/Desktop/gopikrishna$
```